ERROR-CORRECTING CODES

Edward F. Assmus, Jr., et al

Frazyer Research Company

Prepared for:

Air Force Cambridge Research Laboratories

31 August 1972

AFCRL-72-0504

ERROR-CORRECTING CODES

Edward F. Assmus, Jr.
Harold F. Mattson, Jr.

Frazyer Research Company
105 Dorset Road
Syracuse, New York 13210

Contract No. F19628-71-C-0014
Project No. 5628
Task No. 562801
Work Unit No. 56280101

FINAL REPORT

Period Covered: August 1, 1971 through July 31, 1972

August 31, 1972

Contract Monitor: Vera S. Pless
Data Sciences Laboratory

Approved for public release; distribution unlimited

Prepared for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS 01730

14
R

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Frazyer Research Company 105 Dorset Road Syracuse, New York 13210	Unclassified
	2b. GROUP

3. REPORT TITLE

ERROR-CORRECTING CODES

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Scientific Final August 1, 1971 to July 31, 1972 Approved: 7 Nov. '72

5. AUTHOR(S) (First name, middle initial, last name)

Edward F. Assmus, Jr.
Harold F. Mattson, Jr.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
August 31, 1972	49	11

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F19628-71-C-0014	
b. PROJECT, TASK, WORK UNIT NOS. 5628-01-01	
c. DOD ELEMENT 61102F	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) AFCRL-72-0504
d. DOD SUBELEMENT 681305	

10. DISTRIBUTION STATEMENT

A - Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
Tech, Other	Air Force Cambridge Research Laboratories (LR) L.G. Hanscom Field Bedford, Massachusetts 01730

13. ABSTRACT

Two decoding methods for the (48,24) extended binary quadratic-residue code are defined and studied. The first is a majority-logic method using 1,081 parity-checks. The second is a modified majority-logic method using 220 parity-checks and further processing. These methods are compared to an older majority-logic approach which used 4,234 parity-checks. Both fail to correct some of the errors of weights 5 and 6, and the "220-checks" method may fail on some of the weight-4 errors. The second part of the report treats the question of whether a Steiner system is the holding pattern of a linear code. The main result is a theorem that a code holding a Steiner system of type (d-1)-d-2d must have several specified properties.

DD FORM 1 NOV 65 1473

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Error-correcting codes						
Quadratic-residue codes						
Minimum weight						
Majority decoding						
Steiner systems						
t-designs						
Orbits						
Permutation groups						

AFCRL-72-0504

ERROR-CORRECTING CODES

Edward F. Assmus, Jr.
Harold F. Mattson, Jr.

Frazyer Research Company
105 Dorset Road
Syracuse, New York 13210

Contract No. F19628-71-C-0014
Project No. 5628
Task No. 562801
Work Unit No. 56280101

FINAL REPORT

Period Covered: August 1, 1971 through July 31, 1972

August 31, 1972

Contract Monitor: Vera S. Pless
Data Sciences Laboratory

Prepared for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS 01730

TABLE OF CONTENTS

PART I

TWO DECODING METHODS FOR A (48,24) CODE

A. A MAJORITY-LOGIC APPROACH WITH 1,081 PARITY-CHECKS

In [2] we established the possibility of majority-logic decoding of the (48,24) binary extended quadratic residue code by a scheme which used some 4,000-odd parity checks. In the present section we examine the same problem with an approach using fewer parity checks.

The reader will be assumed to be familiar with [1] and [2, Part III]; though for continuity a number of results from these sources will be quoted in what follows, an understanding of the techniques and approaches used there will be helpful.

We established in [2] some basic data about the code. Of course, it has minimum weight 12 and is self-orthogonal. Thus all errors of weight 5 or less are correctable, at least in principle. But since the code is not perfect, there may be other correctable errors. In fact, some 19% of the errors of weight 6 are correctable (i.e., are unique leaders in their cosets), and all other errors of weight 6 or more are not correctable, even in principle.

A basic finding on the code was the distribution of the values of $\lambda_6(E)$ as E ranged over all the 6-sets of coordinate-places. $\lambda_6(E)$ is defined as the total number of "clubs" containing E, where a club is the support of a vector of weight 12 from the code. We

actually calculated λ_6 on the 6-sets of the form $\{0,1,2,x,y,z\}$, finding the following distribution.

Table 1. Number of 6-sets* Containing a Given 3-set

λ_6 =	0	1	2	3	4	5
Not fixed	899	2170	1060	536	50	10
Fixed	3	0	0	12	0	0

"Fixed" means the 6-set is invariant under the 3-subgroup of $PSL_2(47)$ which stabilizes the given 3-set; i.e., the subgroup permuting 0,1, and 2 among themselves permutes x,y, and z among themselves.

From Table 1 it is a simple matter to derive the distribution of λ_6 over all the 6-subsets.

Table 2. Distribution of λ_6

λ_6	
0	2,334,960
1	5,629,848
2	2,750,064
3	1,400,976
4	129,720
5	25,944
	Total $\binom{48}{6}$

Another computer finding was the fact that if E is any 6-subset such that $\lambda_6(E) = 0$, and if p is any point not in E, then

$$\sum_{x \in E} \lambda_6(\{p\} \cup E-\{x\}) = 8.$$

*Really the table gives the number of orbits of these 6-sets under the group of order 3 fixing {0,1,2}.

This fact is explained theoretically in Appendix 2 of this Part I.

We now summarize our previous approach to the decoding problem for this code. There are 4,324 vectors in the code of minimum weight 12 having 1's at a given coordinate place; call it p. Let us call the set of all such 4,324 vectors S. We decode by computing the dot-product of an error vector with each of the vectors of S, tallying the number of parity-check failures obtained thereby. Using the fact that S constitutes a 4-design on the 47 points other than p, and the values of λ_6 calculated on the computer, we found that if a correctable error contained the point p, then the number of parity-check failures would be at least 2,224. Otherwise, an uncorrectable error of weight 6 could be detected; and a correctable error not including p would be shifted cyclically, the dot-products would be computed again, and thus eventually the error (unless it were a single error at ∞) would arrive at p and be discovered.

Our first attempt to reduce the number of parity checks stems from the fact that the minimum-weight vectors of the code fall into 3 orbits under the action of the automorphism group $PSL_2(47)$ of the code. One orbit consists of half the vectors, and each of the other two of one-fourth of the vectors of minimum weight. These orbits are each 3-designs and not 4-designs.

Our approach is to use either of the "short" orbits to generate a set S_1 or S_2 of 1,081 parity-checks covering the point p and to calculate the number of parity-check failures that would result from errors of weight 6 or less.

Here the calculations must rely on computer work even more than before, because S_1 and S_2 are merely 2-designs on the 47 remaining points. To facilitate the computer work and to provide some analytical foundation, we prove a result giving the number of parity-check failures in terms of the λ-function.

LEMMA. Let M be a 0,1 matrix. Let C denote the set of columns of M. For each subset X of C, let $\lambda(X)$ be defined as the total number of rows of M having 1's at all the columns in X. For each subset E of C, and $X \subseteq E$, let $e(X,E)$ be the number of rows of M with 1's at X but with a 0 at some column in Y, for all Y satisfying $X \subsetneq Y \subseteq E$.

Then, for all $E \subseteq C$ we have

$$\lambda(X) = \sum_{X \subseteq Y \subseteq E} e(Y,E) \tag{1}$$

$$e(Y,E) = \sum_{Y \subseteq X \subseteq E} (-1)^{|X|-|Y|} \lambda(X) \tag{2}$$

$$\sum_{\substack{|Y| \text{ odd} \\ Y \subseteq E}} e(Y,E) = \sum_{\phi \neq X \subseteq E} (-2)^{|X|-1} \lambda(X). \tag{3}$$

Proof. The definitions imply (1) immediately. Mobius inversion of (1) on the lattice of subsets of E under containment yields (2), and (3) is a simple consequence of (2). We explain (2) and (3) in more detail:

Let the relation $\leq$ be defined on the subsets of E by the rule $X \leq Y$ iff $X \supseteq Y$. Then the subsets of E together with $\leq$ form a lattice L in which $E \leq X$ for all points X of L.

The functions λ and e, defined at each point of the lattice, are related by equation (1). This is the standard prologue to Mobius inversion (see [5]), the result of which is equation (2). But independently of the general theory one could easily verify (2), starting from (1).

To derive equation (3), we use (2) to find that

$$\sum_{\substack{|Y| \text{ odd} \\ Y \subseteq E}} e(Y,E) = \sum_{\substack{|Y| \text{ odd} \\ Y \subseteq E}} \sum_{Y \subseteq X \subseteq E} (-1)^{|X|-|Y|} \lambda(X). \qquad (4)$$

Since $|Y|$ is odd throughout this sum, the sign is always $(-1)^{|X|-1}$; it remains only to work out how many times a given $X \subseteq E$ appears in our double sum. The answer is that X occurs once for every subset Y of X of odd cardinality. Since there are $2^{|X|}$ subsets of X altogether, and half of them are of odd size, we find that the right-hand side of (4) is equal to

$$\sum_{\phi \neq X \subseteq E} 2^{|X|-1} (-1)^{|X|-1} \lambda(X),$$

where we must of course exclude the possibility of X being empty (because the empty set has no subsets of odd size).

COROLLARY. If M is the incidence matrix of a t-design, for some $t \geq 1$, then (3) simplifies to

$$\sum_{\substack{|Y| \text{ odd} \\ Y \subseteq E}} e(Y,E) = w\lambda_1 - 2\binom{w}{2}\lambda_2 + \ldots + (-2)^{t-1}\binom{w}{t}\lambda_t$$

$$+ \sum_{|X|>t,\ X \subseteq E} (-2)^{|X|-1} \lambda(X), \qquad (3')$$

where $w = |E|$.

Proof. That M is the incidence matrix of a t-design implies that, for all $i \leq t$, $\lambda(X)$ is a constant λ_i for all i-subsets X of C. Thus the terms of (3) can be evaluated up through $|X| \leq t$.

We shall apply this formula (3), or (3'), to all of the decoding methods we treat in this Part. For example, in [2] we worked out formula (3') by ad-hoc methods for the 4-design of all weight-12 vectors with 1's at a given coordinate place p (in this (48,24) code). To calculate the number of parity-check failures on error vectors E not containing the special point p we can simply apply (3'), noting that the λ_i for the 4-design is the λ_{i+1} for the 5-design of all minimum weight vectors. Thus for $|E| = w = 4,5,6$, (3') yields

$$4\lambda_2 - 12\lambda_3 + 16\lambda_4 - 8\lambda_5;\ w = 4$$

$$5\lambda_2 - 20\lambda_3 + 40\lambda_4 - 40\lambda_5 + 16\lambda(E);\ w = 5$$

$$6\lambda_2 - 30\lambda_3 + 80\lambda_4 - 120\lambda_5 + 16 \sum_{x \in E} \lambda\,(E-\{x\}) - 32\lambda(E);\ w = 6.$$

These λ_i's are for the 8; 5-12-48 design. This last formula corrects the one appearing in [2, p.III-3].

Our first approach is to investigate the 3-designs consisting of the two "short" orbits of minimum-weight code vectors. From each orbit we select the 1,081 vectors having 1's at a given place p, and we wish to see how well each of these sets of parity-checks performs in majority decoding. To apply the Corollary of (3') to this set-up, we note that the matrix M involved is now the incidence matrix of a 2-design of type 55; 2-11-47. That is, $\lambda_2 = 55$ and $\lambda_1 = 253$. Thus from (3'), the number of parity-check failures when $p \notin E$ is, for $|E| = w$,

$$253w - 2\cdot 55\binom{w}{2} + \sum_{\substack{X \subseteq E \\ |X| \geq 3}} (-2)^{|X|-1}\lambda(X). \qquad (5)$$

Since we know the code corrects all errors of weight 5 or less and some errors of weight 6, we would like to evaluate the sum (5) for all such errors E.

To begin with, we know already the values of (5) for $w \leq 2$. Thus our next job is to find the distribution of λ over the 3-subsets of the columns of the array. To save effort we observe that it suffices to find λ for the subarray consisting of all 55 vectors having 1's at two more fixed places, since the automorphism group of M is 2-set transitive. This means that we calculate this submatrix of 55 rows and, now, 45 columns, and calculate the column sums to find the distributions of λ on 3-subsets. We did this for both orbits* and reproduce the results here (having divided by 3 because a 3-group acts on the columns of the 55 x 45 submatrix).

Table 3. Frequency-distribution of λ values on 3-sets ($\div$ (47·23 = 1,081))

λ =	7	8	9	10	11	12	13	14
D_6	2	0	0	4	2	3	3	1
A_4	0	2	2	2	4	1	1	3

Thus when we apply the values in Table 3 to (5) we find that an error of weight 3 not including p yields the following numbers of parity-check failures. The entries in the table are the distributions from Table 3 corresponding to the indicated numbers F of parity-check failures; when divided by 15 they give the probability that an error of weight 3 not containing p gives F.

*For a description of the orbits D_6 and A_4 and of how they were derived, see Appendix 1.

Table 4. F = Number of Parity-check Failures for an Error of Weight 3 Not Containing p.

F	457	461	465	469	473	477	481	485
Orbit D_6	2	0	0	4	2	3	3	1
Orbit A_4	0	2	2	[illegible]	4	1	1	3

We summarize our results up to this point:

If E has weight 1 and includes p, then the number F of parity-check failures is 1,081 for both D_6 and A_4.

If E has weight 1 and p is not in E, then F = λ_1 (for the 1,081 x 47 array) = 253 for both D_5 and A_4 (from (3')).

If E has weight 2 and includes p, then F = 1,081 - 253 = 828 for both D_6 and A_4.

If E has weight 2 and does not include p, then F = 396 for both D_6 and A_4 (from (3')).

If E has weight 3 and includes p, then F = 1,081 - 396 = 685 for both D_6 and A_4.

If E has weight 3 and does not include p, then F varies from 457 to 485 according to Table 4.

If E has weight 4 and includes p, then F = 1,081 - (457:4:485) = 624:4:596 with distributional values given by Table 4.

And further, if E has weight 4 and does not include p, then F = 472:4:544 with distributional values given by Table 5.

Table 5. Distribution of Values of F for Errors of Weight 4 Not Including p.

F	D_6	A_4
472	0	1
476	0	0
480	0	0
484	1	2
488	1	0
492	8	6
496	13	9
500	11	15
504	20	17
508	14	23
512	16	14
516	39	28
520	12	20
524	10	10
528	10	13
532	4	2
536	4	3
540	1	2
544	1	0
TOTAL	165	$165 = 1/6\binom{45}{2}$.

(Derivation of Table 5: This was done via an APL program which evaluated (3') for each 2-subset of the columns of the 55 x 45 matrix mentioned earlier. For each such 2-set some auxiliary processing was required, as we now explain. An error of weight 4 not including p is a 5-subset $\{\infty,x,y,z,w\}$, where we assume p is ∞ for convenience. Let $PSL_2(47)$ fix ∞ and map the 2-set $\{x,y\}$

to {0,1}. Then σ takes {z,w} to {a,b} $\subseteq$ {2,...,46}. Now let $\tau \in PSL_2(47)$ fix ∞ and take {z,w} to {0,1}. Then τ takes {x,y} to {c,d} $\subseteq$ {2,...,46}. Now $\sigma\tau^{-1}$ maps {0,1} to {a,b}, and $\tau\sigma^{-1}$ maps {0,1} to {c,d}. Under σ, our 4-set of errors {x,y,z,w} is identified with {0,1,a,b}, and we can calculate (3') in part in the 55 x 45 matrix by evaluating $\lambda(\{0,1,a\})$ and $\lambda(\{0,1,b\})$ for this array. (These are the values of $\lambda(\{x,y,z\})$ and $\lambda(\{x,y,w\})$, not necessarily in that order.) Thus, having chosen {a,b} $\subseteq$ {2,...,46} we find the unique mapping which fixes ∞ and sends {a,b} to {0,1}; this is $(\sigma\tau^{-1})^{-1} = \tau\sigma^{-1}$. We apply this map to {0,1} and get {c,d}. We then can find $\lambda(\{0,1,c\})$ and $\lambda(\{0,1,d\})$, which are in some order the values $\lambda(\{x,z,w\})$ and $\lambda(\{y,z,w\})$. Thus the $\binom{45}{2}$ choices of {a,b} lead to all possible values of (3') on 4-sets not including p. We meet each 4-set twice in this manner, "once for {a,b} and once for {c,d}," and again a 3-group acts on the array, so we have divided the frequenc es by 6.)

We n. tabulate the above summary, including also the valaes c F for errors of weight 5.

Table 6. Number F of Parity-Check Failures for Errors of Weights 5 or less

Error Weight		p in Error		p not in Error	
1	D_6	1,081	D_6	253	
	A_4	1,081	A_4	253	
2	D_6	828	D_6	396	
	A_4	828	A_4	396	
3	D_6	685	D_6	457; 469:4:485	(*)
	A_4	685	A_4	461 :4:485	(*)
4	D_6	624; 612:-4:596	D_6	484 :4: 544	(**)
	A_4	620: -4: 596 (*)	A_4	472;484;492:4:540	(**)
5	D_6	597 :4: 537			
	A_4	609;597;589:-4:541(**)	A_4	477:4:565	

(*) See Table 4 for frequency-distribution.

(**) See Table 5 for frequency-distribution.

Concerning Table 6, the reader will have noted that the entry on the left for an error of weight w is the complement in 1,081 of the entry on the right one line higher.

The last entry in the table, that for errors of weight 5 not including p for the A_4-orbit, was derived by performing the 1,081 parity checks on a sample of the errors of weight 5. The resulting distribution has many entries in common with that on the left, namely 541:4:565. This means that although either of the D_6- or A_4- orbits can correct errors up to weight 4 on a purely majority logic basis, the A_4-orbit will fail to tell whether p is in error on some proportion, roughly 26%,of weight-5 errors, according to our sample distribution, reproduced here:

Number of parity check failures for A_4-orbit on a sample of errors of weight 5

Value	Number of errors
477	3
	0
-	0
489	4
493	3
497	9
501	16
505	32
509	64
513	66
517	73
521	118
525	138
529	196
533	165
537	147
Subtotal	1034

Value	Number of Errors
541	131
545	86
549	71
553	35
557	23
561	14
565	7
Subtotal	367
	1034
	1401

367/1401 = 0.26

The reader will notice that the D_6-orbit has values overlapping in magnitude for errors of weight 4 on the right-hand side of Table 6 and of weight 5 on the left, whereas the A_4-orbit does not. For this reason the D_6-orbit would be less suitable than the other for a purely majority approach to decoding even if it had less overlap between the two sides of Table 6 at weight 5 (a situation that we judge so unlikely that we did not run the D_6-orbit against our sample of 5-errors.)

One further comment: the 26% failure rate on 5-errors might be too pessimistic an estimate. We could set up our majority decoding with the A_4-orbit at a threshold of 569, meaning that we change p if and only if the outcome is 569 or more parity-check failures. This would mean that we would never turn a 5-error into a 6-error, and it is conceivable that for many 5-errors some one or more of the 4 or 5 cyclic shifts of a 5-error having 1 at p would have outcome 569 or more, even if the other shifts gave an outcome of 565 or less. If future investigation showed this to be true, then testing of the A_4-orbit against errors of weight 6 would be in order. The left-hand side of Table 6, if continued to 6-errors, would read 604:-4:516 for the A_4-orbit. We are not optimistic about this, however, because the 569 threshold would fail to change p in some 90-odd percent of 6-errors containing p, presumably on the correctable 6-errors as well as on the noncorrectables.

B. A MODIFIED MAJORITY-LOGIC APPROACH WITH 220 PARITY-CHECKS

We now turn to another approach to decoding this code. We consider the set S of 220 code-vectors of weight 12 which cover a set C of 3 fixed coordinates with 1's. We take these 3 coordinate places to be consecutive under our cyclic shift, e.g., C = {0,1,2}. Our approach will require the calculation of these 220 parity-checks and the recording of the number of parity-check failures on given words. Thus we can derive a table of the various possible outcomes by means of equation (3'), at least in principle. We proceed to do this.

Our array S can be viewed outside of C as the incidence matrix of a 2-design on the 45 coordinates. (Recall that 220 is the value of λ_3 for the 5-design of all weight-12 code-vectors.) As such, λ_1 is 44 and λ_2 is 8. Thus we can easily fill in the entries in the upper left corner of the following table, by use of (3'). Following the table we explain where the other entries have come from.

Table 7. Number of Parity-Check Failures for the 220-Row Array Covering C = 3-set

Weight of Error	Weight of Error on C			
	3	2	1	0
		z	y	x
1			220	44
2		0	176	72
3	220	44	148	D_3 = 84:4:104
4	176	72	$220-D_3$ = 136:-4:116	$D_4 \doteq$ 88:4:116
5	148	D_3	$220-D_4 \doteq$ 132:-4:104	$D_5 \doteq$ 84:4:128
6	$220-D_3$	D_4	$220-D_5 \doteq$ 136:-4:92	$D_6 \doteq$ 88:4:120 $D_6' \doteq$ 84:4:136

The various single-integer entries were all derived from (3'). The D's under column 0 stand for the distributions of the numbers of parity-check failures arising from errors of the indicated weight with 000 on C. All other entries in the table have obviously the indicated distributions in terms of these D's. D_3 is already known from Table 1 combined with (3'). For the other values of D it would have been prohibitive to find the data called for by formula (3'), so we directly sampled the distribution of the number of parity-check failures for each of the cases of error-weight 4,5, and 6; in the latter we did this for the distribution D_6 corresponding to correctable errors of weight 6, and for D_6', the uncorrectables of that weight. The symbol $\doteq$ means here that a small sample has been used to determine the distribution, so we caution the reader that conceivably 80, say, occurs in D_4, or 132 in D_5. (The distribution D_6 here has no relation to the D_6-orbit of Section A.)

We record in Table 8 the distributions of our samples and the sample sizes, reserving a few further comments until after.

Table 8. Distributions D_3 and Samples of D_4, D_5, D_6, D_6'

Number of Parity-Check Failures; Sample Size

	84	88	92	96	100	104	108	112	116	120	124	128	132	136	Total
D_3	2700	6510	3180	1620	150	30									14190
D_4	11	51	178	353	423	300	127	31	5						1479
D_5	2	16	39	136	270	313	318	190	94	18	4	1			1401
D_6		1	7	40	157	336	463	402	181	53					1640
D_6'	4	27	62	118	125	155	261	311	313	172	62	16	13	1	1640

As we said, D_3 is exact. For the later values, we know from (3') that they must be multiples of 4. We can put bounds on D_4 in case the error E_4 with C makes a 7-set with $\lambda_7 = 0$. For then we know from (3') that the outcome must be between 80 and 112, since λ_6 takes only the values 0:1:5. It appears that the true distribution D_6 extends higher than 120. As to the choice of our samples, we have no idea what randomness means here; we chose our sample 4-sets by randomly introducing (via the "deal" instruction of APL) 2 more points into each of the 990 2-subsets of our 45 points and eliminating duplicated 4-sets. We chose our 5-sets by randomly introducing one more point into our 4-sets. The two samples of 6-sets were each chosen as the union of two orbits of appropriate 6-sets under the "$a^2x + b$" subgroup of $PSL_2(47)$ with 6-sets intersecting C eliminated. The assembly-language program calculating the entries in the table was run on an IBM 370.

We now introduce a table giving the probability of occurrence on the binary symmetric channel of errors of the indicated weights in words of length 48 for channel error probabilities of 0.01, 0.005, and 0.001.

Table 9. Probability of Occurence of Errors of Indicated Weight on BSC.

Error Weight	Channel Error Probability 0.01	0.005	0.001
0	0.62	0.79	0.95
1	0.30	0.19	0.046
2	7.1×10^{-2}	2.2×10^{-2}	1.1×10^{-3}
3	1.1×10^{-2}	1.7×10^{-3}	1.7×10^{-5}
4	1.3×10^{-3}	9.8×10^{-5}	1.9×10^{-7}
5	1.1×10^{-4}	4.3×10^{-6}	1.6×10^{-9}
6*	1.5×10^{-6}	3.0×10^{-8}	2.2×10^{-12}

*These entries are for errors of weight 6 which our code can correct.

Having developed Table 7, we now use it to develop a decoding algorithm. Our basic procedure is to perform the 220 parity-checks, record the number of 1's among the outcomes in a 3-stage memory Mem; we then shift the received word, repeat the 220 checks, and enter the outcome in the memory, having shifted the previous outcome into the second of the 3 stages. We repeat this procedure, many times if necessary, so that after the second cyclic shift we always have the three most recent outcomes in our 3-stage memory.

We assume that the circuitry to perform this algorithm can be run at very high speeds compared to the transmission rate used in the channel, so that the received word can be cyclic-shifted as many as 5·47 or even 10·47 times before the next word comes in. It is our understanding that processing speeds of this relative magnitude have been practical for several years already.

Our approach rests on the following simple observation; Every error which this code can correct contains the 5-bit pattern 0001a, where a is 0 or 1, and as this pattern is shifted (to the left) through the three consecutive coordinate positions in our set C, we get the outcomes

x when 000 occupies C

y when 001 occupies C

z when 011 occupies C

y when 010 occupies C

where, in the row of Table 7 corresponding to the weight of our error, x is a value in the column headed 0, y is a value in that headed 1, and z is a value in that headed 2. Thus when 01a occupies C, the 3-stage memory will contain <x,y,z> or <x,y,y>.

Conversely, if the 3-stage memory contains <x,y,z> or <x,y,y>, then if we neglect ambiguities arising from the (rather improbable) overlapping of values between column 0 and column 1 in Table 7, we can conclude that the pattern 0001a gave rise to these values. (We shall discuss the ambiguities later.)

Our decoding procedure will therefore be to change the value in the middle coordinate place of C whenever we see <x,y,z> or <x,y,y> in our 3-stage memory.

We now must analyze whether this procedure can work and if possible, to what extent it fails. That is, since some entries appear in more than one row of Table 7,

- can we distinguish (true) x,y, and z values on one row from (wrong) values on another row? In otner words, how can we tell from the values in Mem = < , , > what row we are on?
- what effect will the overlapping of values between different columns ir a given row have?
- what do we do with an error at the infinite place?
- what is our stopping criterion?

We proceed to take up these questions. Until otherwise specified, the infinite place is assumed to be error-free.

What row are we on? In a nutshell, we can't always tell, but at least at first, it doesn't matter! For example, a single error will yield many times Mem = <44,44,44> in the 3-stage memory, and the weight-3 error 01101 will yield the same at one point in its shifting through C. But the single error will eventually yield <44,220,220>, telling us to correct that error; whereas an error of weight 3 can never yield 220 at two different cyclic shifts.

In effect we have verified the procedure for errors of weight 1. We shall answer the question (of what row we are on) indirectly by showing that if we are on any given row the procedure will decode correctly, except for the problem of ambiguity which we shall discuss later.

Thus an error of weight 2 has as its "change-value" Mem = <72,176,0> or Mem = <72,176,176>. There is no problem of ambiguity at weight 2; we must check, however, whether an error of weight 4 would be incorrectly decoded by the choice of the above two "change-values" for Mem. Since 0 is not in our Table 7 for weight 4, only the value Mem = <72,176,176> might arise from an error of weight 4. But if it does arise, it could (and would) come only from the error 1111; the decoding procedure would change the third 1 from the left to a 0 -- a correct action.

An error of weight 3 would have as "change-values" Mem = <x,148,44> or Mem = <x,148,148>, where x is from distribution D_3 = {84:4:104} with the frequency specified exactly in Table 8. Again there is no problem of ambiguity, so that we need only check whether an error of weight 5 might be incorrectly decoded by this choice of "change-values." The only possible change value is <x,148,148>, since 44 does not arise with weight 5. Such a value of Mem arises from exactly those errors of weight 5 having four consecutive 1's, and the decoding procedure corrects the third of these 1's.

We have shown that for all errors of weight at most 3, our procedure gives correct decoding.

Consider now an error of weight 4. The "change-values" are Mem = <x,y,72> or <x,y,y'>, where x is from distribution $D_4 \doteq$ {88:4:116} and y and y' are from distribution $220\text{-}D_3$ = {136:-4:116}. Some ambiguity is slightly possible in that 116 is a value for both x and y. For example, the "change-value" <116,y,y'> might arise from the error 00100 as it passed through C, yielding the 116 for C = 001. Notice that we would have to begin

reading the contents of Mem for this error right here, or else with high probability we would have reduced this error to weight 3 before we reached this ambiguity. This observation suggests that we might improve the performance of our decoder by ruling that any "change-value" including 116 should not be acted on. Other similar ambiguities can arise to which the same comments apply.

Now consider the effect of these "change-values" on errors of weight 6, which could give rise to Mem = <x,y,y'> in the following cases:

Case 1. If it had four consecutive 1's; again, as in previous cases, our procedure would decode such an error properly.

Case 2. If the weight-6 error is not correctable, then the above change value can arise when 000 is in C; this event would lead to an increase in the error weight.

If the error is correctable, the same possibility has a slight chance of occurring, because $D_6 \doteq \{88{:}4{:}120\}$ contains both 116 and 120 in common with $220\text{-}D_3$.

Case 3. The distribution $220\text{-}D_5 \doteq \{136{:}{-}4{:}92\}$, which occurs with a weight-6 error having weight 1 on C, contains all of $220\text{-}D_3$ and almost all of D_4, so it might lead to incorrect decoding. For example, as 00100 goes through C, the first value, when C = 001, in Mem might be an x in D_4, and the second and third values might be y and y' in $220\text{-}D_3$. At that point the content of C would be 100 and an incorrect action would be taken.

It should also be observed that our change-values chosen for weight 4 will yield correct decoding of some weight-6 errors because D_6 contains D_4, $220\text{-}D_5$ contains

$220-D_3$, and D_4, the z-values for weight 6, overlap $220-D_3$ in 116. That is, some errors of weight 6 will yield values of Mem, as 001a passes through C, equal to the change-values chosen for weight 4.

We summarize the situation at weight 4: Since 116 is an outcome of such low probability, there is high probability that any error of weight 4 will be correctly decoded by our procedure. To estimate the probability numerically would require further study, because we do not know whether the distributions D_4 and $220-D_3$ are sensitive to single cyclic shifts. The option of not taking action in ambiguous cases may lead to an improvement in performance. Some incorrect decoding of errors of weight 6 will result from the method outlined here, but a good estimate of the probability of this event, given an error of weight 6, cannot be given at this time.

Now consider errors of weight 5. We have the following values to consider:

x $D_5 \doteq \{84:4:128\}$
y $220-D_4 \doteq \{132:-4:104\}$
z $D_3 = \{84:4:104\}$.

Here D_5 and $220-D_4$ overlap in at least the 7 values 104, 108,...,128, leading to considerable ambiguity between x- and y- values. Since D_3 is contained in D_5, moreover, there is complete "one-way" ambiguity between z- and x-values, in that no z- value can be distinguished from an x- value. Some correct decoding will take place, but the extent to which this happens would be difficult if not impossible to estimate closely from our present information about the code.

For correctable errors of weight 6, the distributions involved are

x $D_6 \doteq \{88:4:120\}$
y $220-D_5 \doteq \{136:-4:92\}$
z $D_4 \doteq \{88:4:116\}$.

Here the ambiguity is worse, because every x-value except 88 is also a y-value. Still, a judicious choice of change-values might yield correct decoding for some sizable fraction of these errors, but the comments above under the weight-5 case apply here as well.

Reference to Table 9 shows that errors of weights 5 and 6 occur with only a very small probability by comparison to those of higher weights. For this reason the poorer performance of this decoding scheme for errors of those weights might not be a serious drawback to its use on a binary symmetric channel.

What is the stopping criterion? We should stop the procedure after some specified number of cyclic shifts, say 470 (the exact number should be determined by further study.) We examine the value of Mem at this point. If it is <0,0,0>, we say that we have either received a correct word to begin with or have corrected a word received in error. If it is <44,44,44>, we change the contents of the infinite place (if desired) and say that we have corrected a word received in error. If it is anything else we say that we have detected an uncorrectable error.

What do we do about an error at infinity? This question is answered in the preceding paragraph.

C. CONCLUSIONS

The older majority decoding scheme of [2] involved a large number, 4,324, of parity-checks, but it would correctly decode every error this code can correct. In trying to diminish the number of parity checks we have found that, at least by our two methods, we must give up some of the error-correcting ability of the older scheme on the errors of higher weight.

For the future, either of our two methods discussed here may deserve further study in order to determine more precisely their performance on errors of weights 4,5, and 6.

Another future area of study might be the analogue of our "220-checks" method for other extended binary quadratic-residue codes. For these codes are known [1] to yield 3-designs from their minimum-weight vectors; therefore an array of parity-checks covering 3 consecutive places with 1's exists. In general the array is not a 2-design or even a 1-design on the remaining points, so that every entry in the analogue of Table 7 would be a distribution of several values.

DECODING

Appendix 1

The Orbits of $PSL_2(47)$ on the Weight-12 Vectors of the (48,24) Code

In [1, p. 150] we said that there were 3 orbits of the group $PSL_2(47)$ on the weight-12 vectors under consideration, one of length $|PSL_2(47)| \div 6$ and two of half that length. That statement is correct. But we went on to claim that 3 vectors we specified belonged to these respective orbits and to say that both of the short orbits had stabilizer isomorphic to $\sum_3 \times Z_2$, which is isomorphic to the dihedral group D_6 of order 12. As we have found in the course of our current work, these latter claims were partially incorrect. We now set things straight.

Consider the 8 6-sets a,b,...,h, mutually disjoint, of coordinate places of our code defined just as in [1, p. 150] as follows:

a = {0,1,3,16,33,40}	b = {∞,2,13,34,41,43}
c = {4,5,6,18,23,26}	d = {7,12,15,17,29,30}
e = {8,27,32,37,42,45}	f = {9,24,31,36,38,44}
g = {10,11,14,22,39,46}	h = {19,20,21,25,28,35}.

By the notation "ab" we mean the vector of 48 coordinates having 1's at the coordinate places aub and 0's elsewhere. Now let us call the three orbits $\underline{\sigma}$, $\underline{\sigma}_1$, and $\underline{\sigma}_2$. Let $\underline{\sigma}$ stand for the long orbit, with stabilizer isomorphic to $\sum_3$. Then, as before [1, p. 150] ab and df are in $\underline{\sigma}$.

Now, also, ch and eg .re code-vectors, but they are in the same orbit as each other; call it $\underline{c}_1$. The stabilizer of each of these vectors is the same group, isomorphic to D_6.

This corrects our earlier misstatement [1, p. 150] that these last two vectors were in different short orbits. We know that the remaining orbit has a stabilizer of order 12 (see [1, pp. 148-150]), but now we must find it.

Fortunately the groups $PSL_2(q)$ are among the very few for which the set of all subgroups is known. They are described in [3] and more readably in [4]. We must find some subgroup of order 12 of $PSL_2(47)$ stabilizing a code-vector of weight 12 not in the already known orbit $\underline{c}_1$. Thus we begin by listing all the conjugacy classes of groups of order 12 in $PSL_2(47)$, from [4]:

The Subgroups of Order 12 in $PSL_2(47)$

Type	Number of Conjugacy Classes
Cyclic (Z_{12})	1
Dihedral (D_6)	2
Alternating (A_4)	2

Any subgroup of order 12 in $PSL_2(47)$ acts regularly (i.e., without fixed points) on the 48 coordinates, because the only elements of $PSL_2(47)$ fixing any points are of order 23 or 47. Thus any subgroup of order 12 has 4 orbits of length 12 on the 48 coordinates, and each of these orbits can be viewed as a vector of weight 12, which can be tested to see whether it is in our code.

Furthermore, if two subgroups, say H and $\sigma H \sigma^{-1} = H'$, are conjugate in $PSL_2(47)$, then any vector v stabilized by H is in the same orbit as σv stabilized by H'. Thus we need to examine only one group, and the vectors it stabilizes, from each conjugacy class.

We first examined the vectors stabilized by a cyclic group of order 12, finding no codevectors. The element

$$\begin{pmatrix} 27 & 24 \\ 43 & 40 \end{pmatrix}$$

has order 24; its square has order 12.

At this point we can interject a "theological" argument: we know that there are two (48,24) codes of our type, with no vectors in common except the all 0 and all 1, that any element in $PGL_2(47)$ not in $PSL_2(47)$ interchanges these two codes, and that $PSL_2(47)$ is the automorphism group of both codes at the same time, i.e., with the same action.

Now any weight-12 vector in one of the codes with stabilizer H of order 12 is mapped by any element σ of $PGL_2(47)$ not in $PSL_2(47)$ to a vector in the other code with stabilizer $\sigma H \sigma^{-1}$ conjugate in $PGL_2(47)$ to H. This stabilizer is a group of the same type as H, of course, and it is in $PSL_2(47)$. It cannot be conjugate to H in $PSL_2(47)$, however, because if it were, the two codes would have common vectors of weight 12, a contradiction.

From this argument we can deduce that cyclic subgroups of order 12 cannot fix code-vectors (of weights 12 through 36) and that the two short orbits $\underline{\sigma}_1$ and $\underline{\sigma}_2$ in one code are stabilized as follows:

$\underline{\sigma}_1$ by a D_6 and $\underline{\sigma}_2$ by an A_4, with the short orbits in the other code stabilized by the other conjugacy classes of D_6's and A_4's.

The only other a priori possibility, that one group stabilizes both orbits, is ruled out by our computation that the code-vectors ch and eg stabilized by a D_6 are both in the same orbit, and that the other two vectors ad and bf stabilized by this D_6 are not code-vectors.

Thus our job is to find two non-conjugate A_4's. From [4] we learn that inside each D_{24} (a dihedral group of order 48) in $PSL_2(47)$, there are two non-conjugate 4-groups. The normalizers of these 4-groups contain A_4's. We thus produced two such 4-groups and for each one found an element of order 3 in its normalizer, arriving at two non-conjugate A_4's in this way.

In more detail, any 4-group in a dihedral group must contain the rotation χ through 180° as one of its elements; for the two reflections in it commute with each other, which implies they are reflections about perpendicular lines and that their product is χ.

For χ we took

$$\chi = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

We chose

$$\xi = \begin{pmatrix} 13 & 26 \\ 26 & -13 \end{pmatrix} \qquad \xi' = \begin{pmatrix} 13 & -26 \\ -26 & -13 \end{pmatrix}$$

to get two non-conjugate 4-groups $\langle \chi, \xi \rangle$ and $\langle \chi, \xi' \rangle$ in $PSL_2(47)$. Then simply by solving linear equations we found elements σ and σ' of order 3 satisfying

$$\sigma\xi = -\chi\sigma$$
$$\sigma\chi = \xi\chi\sigma$$

and

$$\sigma'\xi' = -\chi\sigma'$$
$$\sigma'\chi = \xi'\chi\sigma'.$$

Our two A_4's are thus $\langle \chi, \xi, \sigma \rangle$ and $\langle \chi, \xi', \sigma' \rangle$. We found that the latter of these two groups did indeed stabilize a code vector, namely, the one with support $\{6,9,12,15,21,25,26,27,38,39,40,43\}$. This vector is a member of the other short orbit $\underline{\sigma}_2$. (Recall that ch and eg are in $\underline{\sigma}_1$.)

DECODING

Appendix 2

We give here a theoretical explanation for Proposition 2 of [2], a special result we obtained in 1970 with a computer. The computer result is for the (48,24) extended binary quadratic-residue code, denoted by A here. Although the theoretical explanation is a general theorem we state it only for this code. The statement and proof will make it obvious how to use the result for other codes.

Recall that the computer result showed that given a 6-subset E of coordinate places of this (48,24) code then, provided that E was not a subset of the support of a minimum-weight code-vector, given any coordinate place p not in E, the number of minimum weight vectors whose support contained p and intersected E in a 5-subset was always 8. I.e., we had a certain uniformity on "correctable" weight-6 errors that could not be explained by the action of $PSL_2(47)$. As we shall see below, the uniformity is explained by the existence of a combinatorial design. Precisely, we have the following.

PROPOSITION. Suppose E is a 6-subset of coordinate places of the extended quadratic-residue (48,24) code A over GF(2) and suppose also that E is not contained in the support of a minimum-weight code-vector. Then, the (42,24) code B obtained by neglecting the 6 coordinate-places of E has a holding pattern which is a 1-design. In particular, the minimum-weight vectors

of B have weight 7 and yield a 1-design of type 8; 1-7-42.

Before proving the Proposition we remark that the minimum weight of the "contracted" code, the (42,24), is in fact 7 since there are weight-12 vectors whose support meets E in a 5-subset. The fact that the holding pattern of the (42,24) is a 1-design is precisely the fact that each coordinate place p not in E is contained in the support of the same number, namely 8, of minimum weight vectors of the (48,24) meeting E precisely in a 5-subset.

Proof. The coordinate places of E correspond to 6 linearly independent functionals since the minimum weight (12) in $A^{\perp} = A$ is greater than 6. Thus, contracting, we have a (42,24) code B whose orthogonal, $B^{\perp}$, consists of the contraction of the subcode of A consisting of those vectors which are 0 over E. The non-zero weights occuring in $B^{\perp}$ are, therefore, 12,16,20,24,28,32 since no vector in $B^{\perp}$ can have weight 36 due to our choice of E. (A weight-36 vector of $B^{\perp}$ would imply that A contained a weight-12 vector whose support contained E.) Thus $B^{\perp}$ has 6 non-zero weights and B has minimum weight 7. Applying Theorem 4.2 of [1] we have the Proposition.

This result has been found independently and in apparently more generality by Delsarte and Goethals [6,7]

BIBLIOGRAPHY

1. E. F. Assmus, Jr., and H. F. Mattson, Jr., "New 5-designs," J. Comb. Theory 6 (1969) 122-151.

2. _____, Algebraic Theory of Codes II, Final Report, Contract No. F19628-69-C-0068, Air Force Cambridge Research Laboratories, AFSC, 15 October 1970.

3. L. E. Dickson, Linear Groups, Dover, 1959.

4. B. Huppert, Endliche Gruppen, Berlin, Heidelberg, New York, Springer, 1967.

5. G. C. Rota, "On the foundations of combinatorial theory I. Theory of Möbius functions," Z. Wahrscheinlichkeitstheorie 2 (1964), 340-368.

6. P. Delsarte, "Four fundamental parameters of a code" Report R 184, MBLE Research Laboratory, Brussels, January 1972.

7. J. M. Goethals, private communication.

PART II

FROM STEINER SYSTEMS TO CODES

We continue in this section a discussion we began in 1967 and reported on in [1, IV, Section 1], where the problem we restate below was first formulated. We treated a particular example of the general problem in 1970, see [2, V, Section 6], and we return to it again here with applications to "t-design decoding" in mind.

The general question is the following: Given a combinatorial design, when can it be realized as the holding pattern of a linear code? Recall that the "holding pattern" of a linear code is the collection of supports of its minimal weight vectors, the "support" of a vector being the set of coordinate places where it is non-zero. For decoding purposes we might want to demand that the linear code be self-orthogonal; we will return to that question later but first we prove the following easy

PROPOSITION 1. Every trivial combinatorial design is the holding pattern of a linear code.

Proof. A trivial design consists of all d-subsets of a given n-set. Thus [3, §4, p. 137] the linear code we seek must be optimal. Let, therefore, F be a finite field with at least n elements and consider the vector space A of polynomials over F of degree less than or equal to n-d. The dimension of A over F is n-d+1 = k. Let $\alpha_1,\ldots,\alpha_n$ be n distinct elements of F and to each polynomial P(x) in A associate the n-tuple

$$(P(\alpha_1), P(\alpha_2),\ldots, P(\alpha_n)) = a_P$$

The set of all such n-tuples (as $P(x)$ varies in A) is clearly a linear subspace of F^n, the space of all n-tuples over F. In fact, the map $P(x) \to a_p$ is a linear transformation of A into F^n and since the α_i's are distinct $a_p = (0,0,\ldots,0)$ implies $P(x)$ is the zero polynomial. Thus, we've constructed an (n,k) code over F (called a shortened Reed-Solomon code), and since each $P(x)$ in A has at most n-d zeroes, the minimum weight is at least $n-(n-d) = d = n-k+1$. Hence equality must hold and the code is optimal with a holding pattern consisting of all d-subsets, as one easily sees directly or via [3, §4, p. 137].

Thus, trivial combinatorial designs correspond to optimal codes. The question of which non-trivial designs are holding patterns of linear codes has no such easy answer as far as we know. As motivation for the following discussion we restate in a slightly more general form a proposition we proved in 1971. (See [4, Part I, §3].)

PROPOSITION 2. Let A be a linear formally self-orthogonal (n,k) code with minimum weight d where n=2d, k=d. Then no two d-subsets which are supports of minimum weight vectors can overlap in d-1 coordinate places.

The proof of this slight generalization is the same as that in the reference cited just above.

Proposition 2 suggests that a (2d,d) formally self-orthogonal code with minimum weight d might have an interesting holding pattern; it suggests the possibility of a holding pattern which is a Steiner System of type (d-1)—d—2d. For d=4 we have the unique extension of the projective plane of order 2 which is indeed the holding pattern of the self-orthogonal (8,4) extended Hamming code over GF(2) and for d=6 we have the Steiner System

associated with M_{12} (the Mathieu group) which is indeed the holding pattern of the self-orthogonal (12,6) extended ternary Golay code.

These two Steiner Systems may well be the only two of type (d-1)—d—2d. These systems were discussed by N.S. Mendelsohn [5] from a purely set-theoretic and combinatorial point of view. Before investigating the possibility of such systems being holding patterns we record the following two elementary propositions.

PROPOSITION 3. Suppose the holding pattern of an (n,k) code A is a Steiner System of type t—d—n with $1 < t < d < n$, and $n-d \geq t$. Then the minimum weight d_0 of $A^{\perp}$, the orthogonal to A, is greater than t; and also, k is greater than t.

Proof. For a t-design (Steiner System or not) the number of blocks containing an i-subset and avoiding a j-subset is independent of the given subsets provided i+j is not greater than t, a fact easily established by contracting on a (t-j)-subset containing the i-subset and using the complementary design. Thus, given any s-subset of coordinate places, where $s \leq t$, there is a block of the Steiner System intersecting the s-subset in a singleton. Thus the s-subset cannot support a vector of $A^{\perp}$. This proves the first assertion. To prove the second, note that the well known bound $d' \leq n-k+1$ for general (n,k) codes of minimum distance d' yields, for $A^{\perp}$, $t+1 \leq d_0 \leq n - (n-k) + 1$, where d_0 is the minimum distance of $A^{\perp}$. Thus $t \leq k$ and equality holds if and only if $A^{\perp}$ is an optimal code with $d_0 = t + 1$. But if $A^{\perp}$ were such an optimal code, we could find a vector of weight $t + 1$ in $A^{\perp}$ the support of which had only one point in common with that of a weight-d vector in A (by our hypothesis $n-d \geq t$). This contradiction proves $t < k$.

PROPOSITION 4. There are unique Steiner Systems of type (d-1)—d—2d for d=1,2,4 and 6. For such a system with d greater than 2 we must have $d \equiv 0,4 \pmod 6$. The first instance of a failure of the obvious necessary number-theoretic conditions is d = 24.

Proof. For d = 1 and 2 the trivial designs meet the requirements and are obviously unique. For d = 4 and 6 we have the classical systems described above which are known to be unique. For d > 2 contracting on d - 3 points gives a Steiner Triple System and hence $2d - (d-3) \equiv 1 \text{ or } 3 \pmod 6$; hence $d \equiv 0 \text{ or } 4 \pmod 6$. One checks easily that for d = 10, 12, 16, 18, and 22 the necessary conditions are met but that for d = 24 they fail. This proves the Proposition.

We now come to the main point of this section. We consider a Steiner System of type (d-1)—d—2d and assume that it is the holding pattern of a linear (2d,k) code and ask for properties of that code. The following theorem shows that such a code must necessarily be highly interesting and, thus, we have theoretical evidence for non-existence -- except, of course, for d = 4 and 6.

THEOREM. Let A be a (2d,k) code whose holding pattern is a Steiner System of type (d-1)—d—2d. Then k = d. Moreover A is formally self-orthogonal and the holding pattern of $A^{\perp}$ is precisely that of A. The weight distribution of A is uniquely determined.

Proof. By Proposition 3 we have $k \geq d$. But $k < 2d - d + 1 = d + 1$ by [3, p. 137], since A cannot be optimal. Thus k = d and the first assertion is proved. Now $A^{\perp}$ is therefore a (2d,d) code also and by Proposition 3 again its minimum weight is at least d but less than or equal to 2d - d + 1 = d + 1. The value d + 1 would imply that $A^{\perp}$ were optimal, which would lead to the same contradiction as at the end of the proof of Proposition 3.

Before showing that the holding pattern of $A^{\perp}$ is that of A we prove that A is formally self-orthogonal (meaning that A and $A^{\perp}$ have the same weight-distribution). From the MacWilliams' Equations we have that

$$dA_d + \binom{2d-1}{d-1} A_{d+1} = (q-1)\binom{2d}{d-1}$$

where A_i denotes the number of vectors of weight i in A.

Since $dA_d = \binom{d}{d-1} A_d = \binom{d}{d-1} (q-1) \frac{\binom{2d}{d-1}}{\binom{d}{d-1}} = (q-1) \binom{2d}{d-1}$,

$A_{d+1} = 0$. Applying [3, Theorem 4.2, p. 138] we find that the holding pattern of $A^{\perp}$ is a t-design with parameters (d-1)—d—2d. But again from the MacWilliams equations we conclude that $\lambda = 1$ and, moreover, that the weight distributions of A and $A^{\perp}$ are the same; i.e., A is formally self-orthogonal.

To finish the proof of the Theorem we need to show that the holding pattern of $A^{\perp}$ is precisely the given Steiner System. But this follows immediately from the fact that in a system with parameters (d-1)—d—2d the complement of a block is a block and no vector v of $A^{\perp}$ and w of A can have their supports intersecting in only a singleton.

REMARK: The unique weight distribution of A (calculable from the MacWilliams equations) begins

$$A_d = (q-1) \frac{\binom{2d}{d-1}}{\binom{d}{d-1}}$$

$$A_{d+1} = 0$$

$$A_{d+2} = (q-1) \binom{2d}{d+2} (q - \tfrac{d}{2})$$

This immediately implies that $q \geq \frac{d}{2}$. Note that the two known systems can be achieved as holding patterns over fields with $q = \frac{d}{2}$. The next case, d = 10, has been handled combinatorially by Mendelsohn and Hung with a computer. Their result shows that there does not exist a Steiner System of type (4,5,15), a fortiori none of type (9,10,20).

Bibliography

1. E. F. Assmus, Jr. and H. F. Mattson, Jr. "Research to develop the algebraic theory of codes," AFCRL-68-0478 Final Report, 14 September 1968; Contract No. AF 19(628)-5998.

2. _____, "Algebraic theory of codes II", AFCRL-71-0013 Final Report, 15 October 1970; Contract No. F19628-69-C-0068.

3. _____, "New 5-designs", Journal of Combinatorial Theory, vol. 6 (1969), 122-151.

4. _____, "Error-correcting codes", AFCRL-71-0561 Scientific Report No. 1, August 31, 1971; Contract No. F19628-71-C-0014.

5. N. S. Mendelsohn, "A theorem on Steiner systems" Can. J. Math. XXII (1970), 1010-1015.